1

SQL Server 2005 Architecture

The days of SQL Server being a departmental database are long gone, and SQL Server can now easily scale to databases dozens of terabytes in size. In this chapter, we lay some of the groundwork that will be used throughout the book. We first discuss how the role of the DBA has changed since some of the earlier releases of SQL Server and then quickly jump into architecture and tools available to you as an administrator. This chapter is not a deep dive into the architecture but provides enough information to give you an understanding of how SQL Server operates.

Growing Role of a DBA

The role of the database administrator (DBA) has been changing slowly over the past few versions of the SQL Server product. In SQL Server 2005, this slow transition of the DBA role has been accelerated immensely. Traditionally, a DBA would fit into one of two roles: development or administration. It's much tougher to draw a line now between DBA roles in SQL Server 2005. As lines blur and morph, DBAs have to quickly prepare themselves to take on different roles. If you don't position yourself to be more versatile, you may be destined for a career of watching SQL Server alerts and backups.

Production DBA

Production DBAs fall into the traditional role of a DBA. They are a company's insurance policy that the production database won't go down. If the database does go down, the company cashes in its insurance policy in exchange for a recovered database. The Production DBA also ensures the server is performing optimally and promotes database changes from development to QA to production.

Since SQL Server 2000, there has been a trend away from full-time Production DBAs, and the role has merged with that of the Development DBA. The trend may have slowed, though, with laws such as Sarbanes-Oxley, where you need a separation of power between the person developing the change and the person implementing the change. Other tasks that a Production DBA does are:

- Install SQL Server instances and service packs
- Monitor performance problems
- Install scripts from development
- Create baselines of performance metrics
- Configure the SQL Server optimally
- Create disaster recovery and scalability plans
- Ensure that backups have been run

In a large organization, a Production DBA may fall into the operations department, which would consist of the network and Windows-support administrators. Placing a Production DBA in a development group removes the separation of power that may be needed for some regulatory reasons. It may create an environment where "rush" changes are immediately put into production, without proper inspection and auditing.

Development DBA

Development DBAs also play a very traditional role in an organization. They wear more of a developer's hat and are the development staff's database experts and representatives. This administrator ensures that all stored procedures are optimally written and that the database is modeled correctly, both physically and logically. He or she also may be the person who writes the migration processes to migrate the database from one release to the next. The Development DBA typically does not receive calls at two in the morning, unless the Production DBA needs to escalate. Other Development DBA roles may be:

- Model an application database
- Create stored procedures
- Develop the change scripts that go to the Production DBA
- Performance-tune queries and stored procedures
- Possibly create any data migration
- Serve as an escalation point for the Production DBA

The Development DBA typically would report to the development group. He or she would receive requests from a business analyst or another developer. In a traditional sense, Development DBAs should never have modification access to a production database. They should, however, have read-only access to the production database to debug in a time of escalation.

Business Intelligence DBA

The Business Intelligence (BI) DBA is a new role that has grown due to the increased surface area of SQL Server. In SQL Server 2005, BI has grown to be an incredibly important feature set that many businesses cannot live without. The BI DBA is an expert at these features. He or she is the one who creates your SSIS packages to perform Extract Transform and Load (ETL) processes or reports for users. In many organizations, the role is so large that the BI DBA functions may be broken into smaller subsets, and you may have specialized DBAs to perform tasks such as SSIS or reports. In the world of SQL Server, a BI DBA is responsible for the following types of functions:

- ❑ Develop data-migration packages
- ❑ Model Analysis Services cubes and solutions
- ❑ Work with the analyst to develop KPI measures for Business Scorecard Manager
- ❑ Create reports using Reporting Services
- ❑ Develop a Notification Services solution
- ❑ Create ETL using Integration Services
- ❑ Develop deployment packages that will be sent to the Production DBA

Organizationally, the BI DBA most often reports to the development group. In some cases with Analysis Services experts, you may see them report to the analyst group or the project management office. In some small organizations, the BI DBA may report directly to an executive such as a CFO.

Hybrid DBA

The most exciting role for a DBA is a hybrid of all the roles we just mentioned. This Hybrid DBA is very typical with smaller organizations but is becoming popular with larger organizations as well. An organization with high turnover may want to spread their investment over many Hybrid DBAs instead of specialized roles.

Organizationally, you may see these DBAs reporting directly to the product organization or to a specialized DBA group. No matter where these DBAs report, each typically has a slate of products that he or she supports and performs every DBA function for that product. Such DBAs should also have adequate backup personnel to reduce the organization's risk if the Hybrid DBA leaves the company. Also, this DBA should never install his or her own changes into production. Ideally, for regulatory reasons and for stability, the DBA's backup DBA should install the change into production. That way, you can ensure that the DBA who installed the script didn't make ad-hoc changes in order to make the change work. We cover much more about this change-management process in Chapter 10.

The only role of a Hybrid DBA that's questionable is development of stored procedures. In most organizations where we see this role, the Hybrid DBA does not develop stored procedures. Instead, he or she creates difficult stored procedures or tunes the ones causing issues. The developer working on the application develops his or her own stored procedures and then provides them to the Hybrid DBA to package and proof. The main reason for this is that the DBA is too taxed for time, working on other functions of the database.

Industry Trends

We'll get into the SQL Server 2005 features momentarily, but you'll notice a trend as you begin to see the list. Feature after feature will require that a DBA become acclimated to a .NET programming language such as C# or VB.NET to remain effective. For example, if you are a DBA trying to debug a performance problem with a CLR stored procedure, you're going to need to know the language the stored procedure is written in to understand the performance problem. Features like Integration Services and Reporting Services are very much tied to expressions, which are variants of VB.NET.

Each new release of SQL Server since 7.0 has required DBAs to know more things that were traditional concerns of developers only, such as XML. With SQL Server 2005, though, there is a leap forward in the knowledge a DBA must have to be effective. Essentially, if you don't know a .NET programming language, you may be stuck as a Production DBA indefinitely. There are still roles for Production DBAs, but even in such roles, you may be less effective.

SQL Server Architecture

In older editions of SQL Server, you had to use many different tools depending on the function you were trying to perform. In SQL Server 2005, the challenge for Microsoft was to avoid increasing the number of management tools while increasing the features and products that ship with SQL Server. They accomplished this by creating one tool for business-intelligence development and another for management of the entire platform, including business intelligence and the database engine. Both of these tools are based on a lightweight version of Visual Studio 2005.

SQL Server envelops a large surface now. It can act as a reporting tool and store your OLAP cubes. It can also perform your ETL services through SQL Server Integration Services. Most people just use SQL Server for its classic use to just store data. SQL Server 2005 can run on Windows XP, 2000, Vista, and Windows Server 2000 and 2003. Tools such as SharePoint and Office quickly integrate on top of SQL Server and can provide an easy user interface (UI) for SQL Server data. This book covers administration on each of these tiers.

Transaction Log and Database Files

The relational database has experienced a number of enhancements in SQL Server 2005 to make it more robust and scalable. As you make changes to a database in SQL Server, the record is first written to the transaction log. Then, at given checkpoints, it is quickly transferred to the data file. This may be why you see your transaction log grow significantly in the middle of a long-running transaction even if your recovery model is set to simple. (We cover this in much more detail in Chapter 18.)

When you first start SQL Server after a stop, it performs a recovery process on each database. This process reads the transaction log for any transaction written to the transaction log but never sent to the data file and rolls it forward onto the data file. Also, any transaction that has not completed will be rolled back. In SQL Server 2005 Enterprise Edition, this process can be done in parallel across all the databases on your instance. Additionally, a fast recovery feature in Enterprise Edition makes databases available after the roll-forward process is complete.

The transaction log's most important purpose is to serve as an exact point in time in case you need to recover your database. Each data-modifying transaction is logged into the transaction log (although this behavior can be minimized if you turn on certain features). If the database becomes corrupt for whatever reason, you could take a transaction log backup and overlay it on top of your full backup, specifying that you wish to recover to the point in time right before the corruption. Corruption is extremely rare since SQL Server 7.0, but protecting against the remote chance of corruption is the DBA's primary job.

A database can consist of multiple file groups that logically group one or multiple database files. Those data files are written into in 8K data pages. You can specify how much free space you wish to be available in each data page with the fill factor of each index. (We go much more into this in Chapter 14.) In SQL Server 2005, you have the ability to bring your database partially online if a single file is corrupt. In this instance, the DBA can bring the remaining files online for reading and writing, and the user receives an error if he or she tries to access the other parts of the database that are offline. (We cover that much more in Chapter 18.)

The most that you can write into a single row is 8K. You are allowed to create a table larger in width than 8K only if there is a chance that it may not hold 8K of data, such as a table that has all varchar columns. If you attempt to write more than 8K to the row, you will receive an error. Also, if you create a table that writes more than 8K of data, you will receive an error.

SQL Native Client

The SQL Native Client is a data-access method that ships with SQL Server 2005 and is used by both OLE DB and ODBC for accessing SQL Server. The SQL Native Client simplifies access to SQL Server by combining the OLE DB and ODBC libraries into a single access method. The access type exposes some of the new features in SQL Server:

- Database mirroring
- Multiple Active Recordsets (MARS)
- Snapshot isolation
- Query notification
- XML data type support
- User defined data types (UDTs)
- Encryption
- Password expiration

In some of these features, you can make the feature work in other data layers such as Microsoft Data Access Components (MDAC), but it will take more work. MDAC still does exist, and you can still use it if you don't need some of the new functionality of SQL Server 2005. If you are developing a COM-based application, you should use SQL Native Client, and if you are developing a managed code application like in C#, you should consider using the .NET Framework Data Provider for SQL Server, which is very robust and includes the SQL Server 2005 features as well.

System Databases

The system databases in SQL Server are crucial, and you should leave them alone most of the time. The only exception to that rule is the `model` database, which allows you to deploy a change like a stored procedure to any new database created. If a system database is tampered with or corrupted, you risk your SQL Server not starting. They contain all the stored procedures and tables needed for SQL Server to remain online.

The Resource Database

New to SQL Server 2005 is the `Resource` database. This database contains all the read-only critical system tables, metadata, and stored procedures that SQL Server needs to run. It does not contain any information about your instance or your databases, because it is only written to during an installation of a new service pack. The `Resource` database contains all the physical tables and stored procedures referenced logically by other databases. The database can be found by default in `C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\mssqlsystemresource.mdf`, and there is only one `Resource` database per instance.

In SQL Server 2000, when you upgraded to a new service pack, you would need to run many long scripts to drop and recreate system scripts. This process took a long time to run and created an environment that couldn't be rolled back to the previous release after the service pack. In SQL Server 2005, when you upgrade to a new service pack or quick fix, a copy of the `Resource` database overwrites the old database. This allows you to quickly upgrade your SQL Server catalog and allows you to roll back a release.

The `Resource` database cannot be seen through Management Studio and should never be altered unless you're under instruction to do so by Microsoft Product Support Services (PSS). You can connect to the database under certain single-user mode conditions by typing the command `USE MSSQLSystemResource`. The majority of what a DBA does is run simple queries against it while connected to any database. For example, if you were to run this query while connected to any database, it would return your `Resource` database's version and the last time it was upgraded:

```
SELECT serverproperty('resourceversion') ResourceDBVersion,
serverproperty('resourcelastupdatedatetime') LastUpdateDate
```

Do not place the `Resource` *database on an encrypted or compressed drive. Doing this may cause upgrade or performance issues.*

The Master Database

The `master` database contains the metadata about your databases (database configuration and file location), logins, and configuration information about the instance. If this important database is lost, your SQL Server may not be able to start. For example, by running the following query, you will see what databases are installed on the server:

```
SELECT * FROM sys.databases
```

The `master` database's role has been slightly diminished in SQL Server 2005 with the addition of the `Resource` database, but it is no less important. The main difference between the `Resource` and `master` databases is that the `master` database holds data specific to your instance, while the `Resource` database

just holds the schema and stored procedures needed to run your instance. You should always back up the `master` database after creating a new database, adding a login, or changing the configuration of the server.

> *You should never create objects in the* `master` *database. If you create objects here, it may cause you to have to make more frequent backups.*

Tempdb Database

The `tempdb` database is like your database's swap file. It's used to hold temporary objects for all logins, and the server may use the database to hold row-version information or system temporary objects. The `tempdb` database is created each time you restart SQL Server. The database will be recreated to be its original database size when the SQL Server is stopped. Since the database is recreated each time, there is no reason to back it up. When you create a temporary object in the `tempdb` database, it writes minimal information into the log file. It is important to have enough space allocated to your `tempdb` database, because many operations that you will use in your database applications use the `tempdb`. Generally speaking, you should set `tempdb` to autogrow as it needs space. If there is not enough space, the user may receive one of the following errors:

- 1101 or 1105: The session connecting to SQL Server must allocate space in `tempdb`.
- 3959: The version store is full.
- 3967: The version store must shrink because `tempdb` is full.

Model Database

`model` is a system database that serves as a template when SQL Server creates a new database. As each database is created, the first step is to copy the objects out of the `model` database and into the empty shell of the new database. The only time this does not apply is when you restore or attach a database from a different server.

> *You can add objects or adjust the settings of the* `model` *database so that any subsequent databases will have those properties set or contain those objects.*

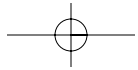
msdb Database

`msdb` is a system database that contains information used by SQL Server agent, log shipping, SSIS, and the backup and restore system for the relational database engine. The database stores all the information about jobs, operators, alerts, and job history. Because it contains this important system-level data, you should back up this database regularly.

Schemas

Schemas enable you to compartmentalize database objects into groups based on their purpose. For example, you may create a schema called `HumanResource` and place all your employee tables and stored procedures into that schema. You could then protect that schema to prevent users from seeing data from within the schema. Think of a schema as a logical grouping of objects within a database.

When you call an object from within a schema, you use a two-part name at a minimum. You may be familiar with the `dbo` schema, which is the default schema for a given database. An `Employee` table in the default



ma is called `dbo.Employee`. This table would be different from `HumanResource.Employee`, if you table in the database. It is a best practice always to refer to a database object by its two-part name, from the `AdventureWorks` database:

```
CT EmployeeID, Salary
 HumanResource.Employee
```

have been around since earlier releases of SQL Server but were not used in the same manner. ly, schemas were tied to your user name. If a DBA were to leave the company, you could not hat DBA's account from SQL Server until you ensured that all the objects inside the DBA's were also moved. That typically created additional development, as you were now pointing all lication to new stored procedure names. This is no longer a problem in SQL Server 2005.

ms

m creates an abstraction layer between the database object and the client. It essentially creates ary logical name for a database object. This abstraction comes in handy when you use linked with linked servers, you have to refer to the four-part qualifier, like the following code:

```
CT Column1, Column2
 LinkedServerName.DatabaseName.SchemaName.TableName
```

g a name creates a usability issue for developers, who at a minimum will receive a massive hand fter typing that long an object name all day long. With synonyms, you can create what equates rector so that anytime someone types `SchemaName.SynonymName`, they're redirected to `ServerName.DatabaseName.SchemaName.Tablename`.

straction layer, synonyms are also useful also if you think you may want to redirect that query to ble or server some day. For example, you may have a table named `Sales2004`, and your syn- me could be `Sales`. When 2005 arrives, you can point the synonym to the new `Sales2005` table.

nonym cannot reference another synonym.

ic Management Views

```
SELECT login_name, COUNT(session_id) as NumberSessions
FROM sys.dm_exec_sessions GROUP BY login_name
```

In fact, DMVs are also sometimes functions and accept parameters. For example, the following code uses the `sys.dm_io_virtual_file_stats` dynamic management function (we use the term *DMV* for simplicity throughout this book) to retrieve the I/O statistics for the AdventureWorks data file.

```
SELECT * FROM
sys.dm_io_virtual_file_stats(DB_ID('AdventureWorks'),
FILE_ID('AdventureWorks_Data'))
```

We cover much more about DMVs throughout this book, starting in Chapter 4.

SQL Server 2005 Data Types

As you create a table, you must assign a data type for each column. In this section, we cover some of the more commonly used data types in SQL Server. Even if you create a custom data type, it must comply with the standard SQL Server data types in some way. For example, you may have created a custom data type (`Address`) by using the following syntax, but notice that it still has to fit inside the `varchar` data type.

```
CREATE TYPE Address
FROM varchar(35) NOT NULL
```

If you are changing the data type of a column in a very large table in SQL Server Management Studio's table designer interface, the operation may take a very long time. The reason for this can be observed by scripting the change from the Management Studio interface. Management Studio creates a secondary temporary table that has a name like `tmpTableName` and then copies the data into the table. Finally, the interface deletes the old table and renames the new table with the new data type. There are other steps along the way, of course, to handle indexes and any relationships in the table.

If you have a very large table with millions of records, this process can take more than ten minutes and in some cases more than hour. To avoid this, you can use a simple one-line T-SQL statement in the query window to change the column's data type. For example, to change the data type of the `Title` column in the `Employees` table to a `varchar(70)`, you could use the following syntax.

```
ALTER TABLE HumanResources.Employee ALTER COLUMN Title Varchar(70)
```

When you convert to a data type that may be incompatible with your data, you may lose important data. For example, if you convert from a numeric data type that has data such as 15.415 to an integer, the number 15.415 would be rounded to a whole number.

Oftentimes, you may wish to write a report against your SQL Server tables to output the data type of each column inside the table. There are dozens of ways to do this, but one method we often see is to join the `sys.objects` table with the `sys.columns` table. There are two functions that you may not be familiar with in the following code. The `type_name()` function translates the data type id into its proper name. To go the opposite direction, you could use the `type_id()` function. The other function of note is `schema_id()`, which is used to return the identity value for the schema. This is mainly useful when you wish to write reports against the SQL Server metadata.

```
SELECT o.name AS ObjectName,
       c.name AS ColumnName,
       TYPE_NAME(c.user_type_id) as DataType
FROM   sys.objects o JOIN sys.columns c
ON     o.object_id = c.object_id
WHERE  o.name = 'Department'
and o.Schema_ID = schema_id('HumanResources')
```

This code returns the following results (note that the `Name` data type is a user-defined type):

```
ObjectName           ColumnName     DataType
---------------------------------------------------
Department           DepartmentID   smallint
Department           Name           Name
Department           GroupName      Name
Department           ModifiedDate   datetime
```

Character Data Types

Character data types include `varchar`, `char`, `nvarchar` and `nchar`, `text`, and `ntext`. This set of data types store character data. The primary difference between the `varchar` and `char` types is data padding. If you have a column called `FirstName` that is a `varchar(20)` data type and you store the value of "Brian" in the column, only five bytes will be physically stored. If you store the same value in a `char(20)` data type, all 20 bytes would be used.

If you're trying to conserve space, why would you ever use a `char` data type? There is a slight overhead to using a `varchar` data type. If you are going to store a two-letter state abbreviation, you're better off using a `char(2)` column. In some DBAs' eyes, this may be a religious conversation, but generally speaking, it's good to find a threshold in your organization and set a small mental standard that anything below this size will become a `char` versus a `varchar`. Our guideline is that, in general, any column that is less than or equal to eight bytes should be stored as a `char` data type instead of a `varchar` data type. Beyond that point, the benefit of using a `varchar` begins to outweigh the cost of the overhead.

The `nvarchar` and `nchar` data types operate the same way as their `varchar` and `char` sister data types, but these data types can handle international Unicode characters. This comes at a cost though. If you were to store the value of "Brian" in an `nvarchar` column, it would use ten bytes, and storing it as an `nchar(20)` would use 40 bytes. Because of this overhead and added space, do not use Unicode columns unless you have a business or language need for them.

The last data types to mention are `text` and `ntext`. The `text` data type stores very large character data on and off the data page. You should use these sparingly, as they may affect your performance. They can store up to 2GB of data in a single row's column. Instead of using the text data type, the `varchar(max)` type is a much better alternative because the performance is better.

Numeric Data Types

Numeric data types consist of `bit`, `tinyint`, `smallint`, `int`, `bigint`, `numeric`, `decimal`, `money`, `float`, and `real`. All of these data types store different types of numeric values. The first data type, `bit`, stores only a 0 or 1, which in most applications translates into true or false. Using the `bit` data type is perfect for on and off flags, and it occupies only a single byte of space. Other common numeric data types are shown in the following table.

Data Type	Stores	Storage Space
`Bit`	0 or 1	1 byte
`Tinyint`	Whole numbers from 0 to 255	1 bytes
`Smallint`	Whole numbers from –32,768 to 32,767	2 bytes
`Int`	Whole numbers from –2,147,483,648 to 2,147,483,647	4 bytes
`Bigint`	Whole numbers from –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
`Numeric`	Numbers from $-10^{38}+1$ through $10^{38}-1$	Up to 17 bytes
`Decimal`	Numbers from $-10^{38}+1$ through $10^{38}-1$	Up to 17 bytes
`Money`	–922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
`Smallmoney`	–214,748.3648 to 214,748.3647	4 bytes

Numeric data types, such as `decimal` and `numeric`, can store a variable amount of numbers to the right and left of the decimal place. *Scale* refers to the amount of numbers to the right of the decimal. *Precision* defines the total size of the number, including the digits to the right of the decimal place. So 14.88531 would be a `numeric(7,5)` or `decimal(7,5)`. If you were to insert 14.25 into a `numeric(5,1)` column, it would be rounded to 14.3.

Binary Data Types

Binary data types such as `varbinary`, `binary`, `varbinary(max)`, or image store binary data such as graphic files, Word documents, or MP3 files. The `image` data type stores up to 2GB files outside the data page. The alternative to an image data type is the `varbinary(max)`, which can hold more than 8K of binary data and generally performs slightly better than an `image` data type.

XML

When XML first came out, developers began to store this hierarchical data into a `text` or `varchar` column. You would typically store data in XML in a database when the columns in the application would be variable, such as a survey application. This wasn't optimal, as you can imagine, because you can't index this type of data inside a text column. In SQL Server 2005, you have the option to store XML data into a proper XML data type that can be indexed, and schema can now be enforced. (We cover much more about these in Chapter 15.)

DateTime

The `datetime` and `smalldatetime` types both store the date and time data for a value. The `smalldatetime` is 4 bytes and stores from January 1, 1900 through June 6, 2079 and is accurate to the nearest minute. The `datetime` data type is 8 bytes and stores from January 1, 1753 through December 31, 9999 to the nearest 3.33 millisecond.

Unfortunately, there is no `date` or `time` data type. If you wish to store just the date, the time of midnight will be time-stamped on each record. If you wish to insert just the time, today's date is implicitly

inserted. To get only the date out of a `datetime` data type, you must essentially "fool" the data type by converting it:

```
SELECT CONVERT(varchar, GetDate(), 101)
```

CLR Integration

In SQL Server 2005, you can also create your own data types and stored procedures using CLR (Common Language Runtime). This allows you to write more complex data types to meet your business needs in Visual Basic or C#, for example. (We cover the administration aspect of these much more in Chapter 8.)

Editions of SQL Server

With SQL Server 2005, there are numerous editions of the SQL Server product. The features available to you in each edition vary widely. The editions you can install on your workstation or server also vary based on the operating system. The editions of SQL Server range from SQL Express on the lowest end to Enterprise Edition on the highest. The prices of these also vary widely, from free to more than $20,000 per processor.

SQL Express

SQL Express is the free version of SQL Server meant for installation to laptops or desktops to support distributed applications such as a remote sales force application. You can use this edition to store sales or inventory data for your disconnected sales force and replicate updated data to them when they become connected again. SQL Express was called Microsoft Desktop Edition (MSDE) in SQL Server 2000. It is extremely lightweight and does not occupy much hard drive space. Vendors are free to distribute SQL Express, and it can be wrapped into your application's installation as just another component.

SQL Express is not meant to scale past a few users. Key features missing from SQL Express are SQL Agent and some of the robust management tools. It does ship with a very lightweight tool for managing the database, but scheduling of backups will have to be done in the Windows scheduler, not SQL Server.

Workgroup and Standard Editions

The Workgroup Edition of SQL Server is the lowest-cost edition of SQL Server editions that you pay for. It scales minimally up to two processors and 3GB of RAM, but it's adequate for small and medium-sized businesses. This edition of SQL Server was initially introduced to compete with lower-end vendors such as MySQL.

The Standard Edition of SQL Server has been beefed up in SQL Server 2005. It now has high-availability options that were exclusive to the Enterprise Edition in SQL Server 2000. For example, you can now cluster SQL Server 2005 Standard Edition instances.

Enterprise, Evaluation, and Developer Editions

Enterprise Edition is the best option for SQL Server if you need to use some of the more advanced business intelligence features or if the uptime of your database is very important. Although the Standard

Edition of SQL Server allows you to have high-availability options, Enterprise Edition far outdoes its sister edition with higher-end clustering as well as more advance mirroring and log-shipping options. The counter to this, of course, is the price. This edition of SQL Server will cost you more than $20,000 per processor if you choose that licensing model. (We discuss licensing later in this chapter.)

The Evaluation Edition of SQL Server is a variant of SQL Server Enterprise Edition that expires after a given time. After the allotted evaluation period, SQL Server will no longer start. The Developer Edition of SQL Server allows you to run all the Enterprise Edition features in a development environment. Neither of these editions is licensed for production use.

Operating System

The edition of SQL Server that you can install varies widely based on the operating system on your server or workstation, as summarized in the following table.

Operating System	SQL Express	Workgroup	Standard	Developer	Enterprise
Windows 2000 Professional (with SP4+)	✓	✓	✓	✓	
Windows 2000 Server (with SP4 +)	✓	✓	✓	✓	✓
Windows 2003 Server (SP1+)	✓	✓	✓	✓	✓
Windows XP Home Edition (with SP2+)	✓			✓	
Windows XP Professional Edition (with SP2+)	✓	✓	✓	✓	✓

Maximum Capacity of SQL Server

Memory and the number of processors is a huge contributing factor when you're scaling SQL Server. As you can imagine, the amount of memory you can scale and the number of processors will vary based on the edition of SQL Server you purchase. In some cases, your scalability is restricted only to the operating system's maximum memory or number of processors. This is where 64 bit becomes really useful. (We cover 64-bit scalability much more in Chapter 15.)

Capacity	SQL Express	Workgroup	Standard	Enterprise
Memory Supported 32 bit	1GB	3GB	OS Maximum	OS Maximum
Memory Supported 64 bit	N/A	N/A	OS Maximum	OS Maximum
Maximum Database Size	4 GB	No Limit	No Limit	No Limit
Number Processors	1	2	4	Non Limit

Database Features by Edition

The main benefit from one edition of SQL Server to the next are the features enabled. In the following set of grids, you can see how the features line up to each other across the various editions. These grids do not capture all the features of SQL Server but instead focus on areas that we receive common questions about and areas that help distinguish the editions.

Developer Features by Edition

SQL Server 2005 really tries to appeal to the developer. It has many features that will improve the efficiency of the developer's day-to-day job or make his or her code more reliable. In most cases, you can see that the developer slate of features works across all the various editions.

Feature	SQL Express	Workgroup	Standard	Enterprise
CLR Integration	✓	✓	✓	✓
XML Data Type	✓	✓	✓	✓
Try...Catch Exception Handling	✓	✓	✓	✓
Service Broker	Client Only	✓	✓	✓

Business Intelligence Features by Edition

The fastest-growth area for SQL Server revolves around business intelligence. Business intelligence allows you to bring the large amounts of data you have in your systems to the mass of users that would normally not touch this data.

Feature	SQL Express	Workgroup	Standard	Enterprise
Analysis Services (SSAS)			Up to 16 instances	Yesup to 50 instances
SSAS Clustering			2 nodes	✓
Parallelism for Data Mining Model Processing				✓
SSAS Data Mining			✓	✓
SSAS Perspectives				✓
SSAS Translations				✓
SSAS Proactive Caching				✓
Partitioned Cubes				✓
Reporting Services	✓	✓	✓	✓
Report Caching			✓	✓
Report Scheduling			✓	✓

Feature	SQL Express	Workgroup	Standard	Enterprise
Report Subscriptions			✓	✓
Data-Drive Subscriptions				✓
Report Builder		✓	✓	✓
Report Manager		✓	✓	✓
Infinite Drill-down				✓
Notification Services			✓	✓
SQL Server Integration Services (SSIS)		✓	✓	✓

DBA Features by Edition

The DBA features have the largest disparity among editions. You can see in the following table that most of the disparity revolves around high availability. If it's of the utmost importance that your system remain available all the time, use Enterprise Edition.

Feature	SQL Express	Workgroup	Standard	Enterprise
Indexed Views	✓	✓	✓	✓
Indexing of XML Data Type		✓	✓	✓
Failover Clustering			2-node	✓
Number of Instances Supported	16	16	16	50
Log Shipping		✓	✓	✓
Database Snapshots				✓
Database Mirroring Primary or Secondary			Safety Full Mode Only	✓
Dedicated Administrator Connection		✓	✓	✓
Dynamic AWE				✓
Fast Start of the Instance				✓
Hot Memory Addition				✓
Online Indexing Operations				✓
Online Page and File Restoration				✓
Replication	Subscriber Only	✓	✓	✓
Table partitioning				✓
Updatable Partitioned Views				✓

Licensing

Every DBA has probably received a dreaded licensing question or two, and we hope to answer some of those common questions in this section. There are several ways to license SQL Server, and we can't address this ever-changing landscape completely in this book. Instead, we've tried to answer common questions that are not as likely to change from year to year. If you were to purchase any of the licenses we refer to in this section, they are compatible with previous releases of SQL Server as well as SQL Server 2005.

The Server plus User Client Access License (CAL) licensing model works well if you can trace each connection to a user and if you have a low number of connections to your SQL Server services. This license licenses the server and each named user connecting to SQL Server.

The Server plus Device CAL licensing model works well if you expect that a moderate number of named devices will connect to your instance. In this model, you license the server and then each device (a kiosk or desktop, for example) connecting to the services of SQL Server. If you have multiple users using a single desktop, you need only a single device CAL.

The Processor licensing model works well if you expect to have a high number of connections on your SQL Server or if you can't identify connections, such as an Internet application exposed to others outside your company. This model licenses each physical or virtual processor available to the server. If you were to disable a processor to the operating system and in turn SQL Server, this processor would not have to be licensed. Once the available processors are licensed, you can have unlimited connections to the server.

Modern Processor Issues

In 2005, Microsoft clarified its licensing stance on multiprocessor systems. Hyperthreading allows a single processor to simulate multiple processors. If you had a four-processor server, you would actually see eight processors in Task Manager. With SQL Server, if you were licensing per processor, you would only have to license the physical chip connected to the mainframe and would not be charged for the hyperthreaded processors. This also applies in a multicore server. In a multicore machine, you would have one physical chip connected to the mainframe that had multiple processors sitting on it. It's essentially a processor hub. You are charged only for a single chip versus each processor on the chip.

Scaling and High Availability Licensing Issues

As we mentioned earlier, you are only charged for the physical chips on the machine if you choose the per-processor model. If you have 10 instances of SQL Server on a single server, you're not charged for each instance in a per-processor model. Another common question is with clusters. In an active-passive cluster, you are only charged for the active server, and the passive server is at no charge. In an active-active cluster, you are charged for each active node, so you might be charged for each node.

Oftentimes, DBAs decide to scale out SSRS, SSIS, or SSAS to avoid slowing down the relational engine. If you were to scale one of the SQL Server BI products off of the SQL Server machine, you would need to license the other server even though SQL Server may not be installed on it.

Summary

In this chapter, we covered the basic architecture for how SQL Server stores its data and how it communicates. We also addressed the various data types and when to use one over another data type. Last, we answered some of the many dreaded SQL Server edition and licensing questions that we often hear from users. Now that we have the groundwork down, we're ready to jump into installing SQL Server and some of the common issues that you may experience.